MEMORANDUM
RM-4783-PR
FEBRUARY 1966

INTEGER PROGRAMMING BY IMPLICIT ENUMERATION AND BALAS' METHOD

Arthur M. Geoffrion

PREPARED FOR:
UNITED STATES AIR FORCE PROJECT RAND

The RAND Corporation
SANTA MONICA • CALIFORNIA

MEMORANDUM
RM-4783-PR
FEBRUARY 1966

INTEGER PROGRAMMING BY IMPLICIT ENUMERATION AND BALAS' METHOD

Arthur M. Geoffrion

This research is sponsored by the United States Air Force under Project RAND—Contract No. AF 49(638)-1700—monitored by the Directorate of Operational Requirements and Development Plans, Deputy Chief of Staff, Research and Development, Hq USAF. Views or conclusions contained in this Memorandum should not be interpreted as representing the official opinion or policy of the United States Air Force.

The RAND Corporation
1700 MAIN ST • SANTA MONICA • CALIFORNIA • 90406

PREFACE

Integer linear programming has been recognized since the early days of linear programming as a class of problems of the greatest practical importance. A number of difficult Air Force problems of basically combinatorial character--such as those that occur in flight scheduling, multi-stage machine scheduling, and maintenance operation sequencing--can be formulated as integer linear programs.

The fact that no computationally satisfactory algorithm yet appears to be known for a sufficiently wide class of such problems causes interest to run high when new approaches are suggested. Such was the case with Balas' work when it first became known in this courtry. This Memorandum presents an elementary technical exposition of a reformulation of Balas' algorithm. It is more economical in terms of computer storage than the original, and forms the basis for a computer routine that has been used to experimentally assess computational efficiency.[4]

The author is a consultant for The RAND Corporation.

SUMMARY

This Memorandum presents a reformulation of the essentials of Balas' algorithm for the zero-one integer linear programming problem, and is based upon the idea of "elementary tree search" that has also been used by Glover as the basis for his multiphase-dual algorithm. The present reformulation requires considerably less computer storage than the original version, and clarifies the rationale behind the algorithm, thereby leading naturally to variants and extensions.

CONTENTS

I. INTRODUCTION

In a recent comprehensive survey[1] of integer programming, M. L. Balinski expressed a belief, "...that various clever methods of enumeration can be the most efficacious means existent by which to obtain solutions to practical problems." The adjective "clever" is undoubtedly meant to imply, of course, that one should do better than complete enumeration. This can be done by using strategies which lead, as the enumeration proceeds, to the generation of information that can be used to exclude large numbers of solutions from further consideration. The exclusion of solutions, which suggests the descriptive term "implicit enumeration" for such methods, can usually be accomplished in a variety of ways--for example, by exploiting the availability of a gradually improving bound on the optimal value of the objective function as better and better feasible solutions are found. It is unfortunate that the exclusion of solutions sometimes gives rise to very large information storage requirements in order to "remember" which solutions have been excluded as well as enumerated, thereby adding another dimension to the need for "cleverness." A computationally successful algorithm must combine a sufficiently high "rate of exclusion" with sufficiently modest storage requirements.

Since Balinski's survey was written, additional support for the viewpoint quoted above has derived from contributions by Balas[2] and Glover,[3] among others. Their work suggests that it may be possible to combine a high rate of exclusion with low storage requirements for a wide class of important problems, namely linear programs that require bounded integer solutions. Such an approach requires only addition for

arithmetic operations, for no systems of linear equations need to be solved, and shares with other implicit enumeration approaches the important advantage that a feasible solution (hopefully a good one) is usually in store if the calculations are stopped prior to natural termination by exhaustion.

The primary purpose of this Memorandum is to give an expository derivation of the main ideas that motivate and underlie the approaches of Balas and Glover, and those that will inevitably follow them. We do not attempt a critique or thorough discussion of their work, but rather draw freely upon their ideas as we develop fundamentals.

In Sec. II we elaborate upon an idea of Glover[3] in order to develop a basic enumerative scheme which achieves a remarkable degree of flexibility with only modest storage requirements. Using this enumerative scheme, in Sec. III we synthesize what is essentially a reformulation of Balas' algorithm. We attempt to make it clear that the algorithm presented here is but a particular member of an entire class of possible algorithms. An example is presented in Sec. IV, and some further remarks are made in the final section. We conclude this introduction with a precise statement of the problem and some preliminary definitions.

Any bounded integer linear programming problem can be written in the form

$$(P) \qquad \underset{x}{\text{Minimize}}\ cx \text{ subject to } b + Ax \geqq 0, \quad x_j = 0 \text{ or } 1,$$

where c is an n-vector, b and 0 are m-vectors, A is an m by n matrix,

and x is a binary n-vector to be chosen.* Any binary x will be called a solution. A solution that satisfies the constraints $b + Ax \geq 0$ will be called a feasible solution, and a feasible solution that minimizes cx over all feasible solutions will be called an optimal feasible solution.

---

*A bounded problem is one for which an upper bound ν_j is available for each variable. The substitution

$$x_j = \sum_{i=0}^{k} 2^i y_{ji},$$

where k is the smallest integer such that $\nu_j \leq 2^{k+1}-1$, y_{ji} binary, permits a binary representation for x_j.

II. A PROCEDURE FOR IMPLICIT ENUMERATION

Since there is a finite number 2^n of solutions, exhaustive enumeration provides a finitely convergent procedure for discovering an optimal feasible solution of (P). As indicated above, not all solutions are to be explicitly enumerated, of course, but rather implicitly enumerated by considering groups of solutions together. To explain how groups of solutions will be defined, we require the notion of a partial solution. A partial solution S is defined as an assignment of binary values to a subset of the n variables. Any variable not assigned a value by S is called free. We adopt the notational convention that the symbol j denotes $x_j = 1$ and the symbol -j denotes $x_j = 0$. Hence, if n = 5 and S = {3,5,-2}, then $x_3 = 1$, $x_5 = 1$, $x_2 = 0$, and x_1 and x_4 are free. It will be seen that the order in which the elements of S are written will be used to represent the order in which the elements are generated. A completion of a partial solution S is defined as a solution that is determined by S together with a binary specification of the values of the free variables. In the above example, there are four possible completions of S:

(0,0,1,0,1),
(0,0,1,1,1),
(1,0,1,0,1), and
(1,0,1,1,1).

Thus, a partial solution S with s elements, say, determines a group of 2^{n-s} different completions or solutions. When there are no free variables, there is only one completion of S, the trivial one determined by S itself.

Implicit enumeration involves generating a sequence of partial solutions and simultaneously considering all completions of each. As the

calculations proceed, feasible solutions are discovered from time to time, and the best one yet found is kept in store as an incumbent. Now it may happen that for a given partial solution S we can determine a best feasible completion of S, i.e., a feasible completion that minimizes cx among all feasible completions of S. If such a best feasible completion is better than the best known feasible solution (assuming that one is known), then it replaces the latter in store. Or we may be able to determine that S has no feasible completion better than the incumbent. In either case, we shall say that we can fathom S.* All completions of a fathomed S have been implicitly enumerated in the sense that they can be excluded from further consideration--except, of course, a best feasible completion of S that unseats the incumbent.

Leaving aside until the next section the important question of how one fathoms a given S, we shall give a flexible procedure for generating a sequence of partial solutions that is non-redundant and terminates only after all 2^n solutions have been enumerated. By non-redundant, we mean that no completion of a partial solution in the sequence ever duplicates a completion of a previous partial solution that was fathomed.

Consider the following scheme for generating a non-redundant sequence $< S^\nu >$ that will terminate only after all 2^n solutions have been (implicitly) enumerated. Start with $S^o = \phi$, where ϕ indicates an empty set. If S^o can be fathomed, we are finished--either there is no feasible solution, or there is one and the best feasible solution can be found.

---

*To cover the case where there are no free variables, we shall agree that such an S is fathomed.

If S^o cannot be fathomed, augment it by specifying a binary value for one additional free variable at a time, each time trying to fathom the resulting partial solution, until at some trial k_1, S^{k_1} is fathomed. Now to be sure of having enough information in the future to enable us to know when all 2^n solutions have been accounted for, we must store* S^{k_1}; and to be sure of having a non-redundant sequence $< S^\nu >$ from $\nu = k_1 + 1$ on, it is obviously necessary and sufficient to have in all future S^ν at least one element complementary to one in S^{k_1}. We may accomplish the storage of S^{k_1} and heed the condition for the non-redundancy of S^{k_1+1}, at least, by taking S^{k_1+1} to be exactly S^{k_1} with its last element multiplied by -1 and underlined. The underline commemorates the fathoming of S^{k_1} (an example is presented below to make these ideas more concrete).

If S^{k_1+1} can be fathomed, then it is easy to see that all completions of S^{k_1} without its last element have been enumerated, and thus that we can "forget" the fathoming of S^{k_1} and of S^{k_1+1} and "remember" only the fact that S^{k_1} without its last element has been fathomed. For example, if $k_1 = 3$ and $S^3 = \{3,5,-2\}$ was fathomed and then $S^4 = \{3,5,\underline{2}\}$ was fathomed, then all completions of $\{3,5\}$ have been accounted for, since the completions of $\{3,5,-2\}$ and $\{3,5,2\}$ dichotomize those of $\{3,5\}$. Thus, fathoming S^3 and S^4 is equivalent to fathoming $\{3,5\}$. Opportunities such as this to forget some history lead to the economical storage

---

*Since the augmentation of S^o was entirely arbitrary, it seems unlikely that we could store the information that all completions of

$$S^{k_1}$$

have been fathomed any more economically than by storing

$$S^{k_1}.$$

This need not be true if we had used a simple rigid rule for generating $< S^\nu >$, for then we would only have to store the rule and the value k_1.

requirements of the procedure we are now motivating. The same motive that directed our choice of S^{k_1+1} directs us to choose S^{k_1+2} as S^{k_1} less its last element with its next to last element multiplied by -1 and underlined. In our hypothetical example, S^5 would be taken to be $\{3,-\underline{5}\}$. Note that S^5 contains an element complementary to one that appears in both previously fathomed partial solutions.

If, on the other hand, S^{k_1+1} cannot be fathomed, then one would augment it by specifying a binary value for one additional free variable at a time, each time trying to fathom the resulting partial solution, until at some later trial k_2, S^{k_2} is fathomed. Note that the sequence $S^{k_1+1}, \ldots, S^{k_2}$, is non-redundant because each contains the complement of an element of S^{k_1}. When S^{k_2} is fathomed, it, in addition to S^{k_1}, must be stored; and every succeeding S^{ν} must contain not only an element that is the complement of one in S^{k_1}, but also one that is the complement of an element of S^{k_2}. Both ends may be accomplished economically by taking S^{k_2+1} as S^{k_2} with its last element complemented and underlined. In our example, if S^4 could not be fathomed and S^5 were $\{3,5,\underline{2},1\}$, say, that could be fathomed ($k_2 = 5$), then S^6 would be taken to be $\{3,5,\underline{2},-\underline{1}\}$.

Continuing along these lines, one is led to the procedure of Fig. 1. In this figure, the content of Steps 1 and 2 is deliberately unspecified in order to leave maximum flexibility in the design of an algorithm by having a general convergence proof. It is important to note that the mechanism by which Step 1 attempts to fathom a partial solution can be as weak or as powerful as desired--so long as S is truly fathomed when it purports to be.

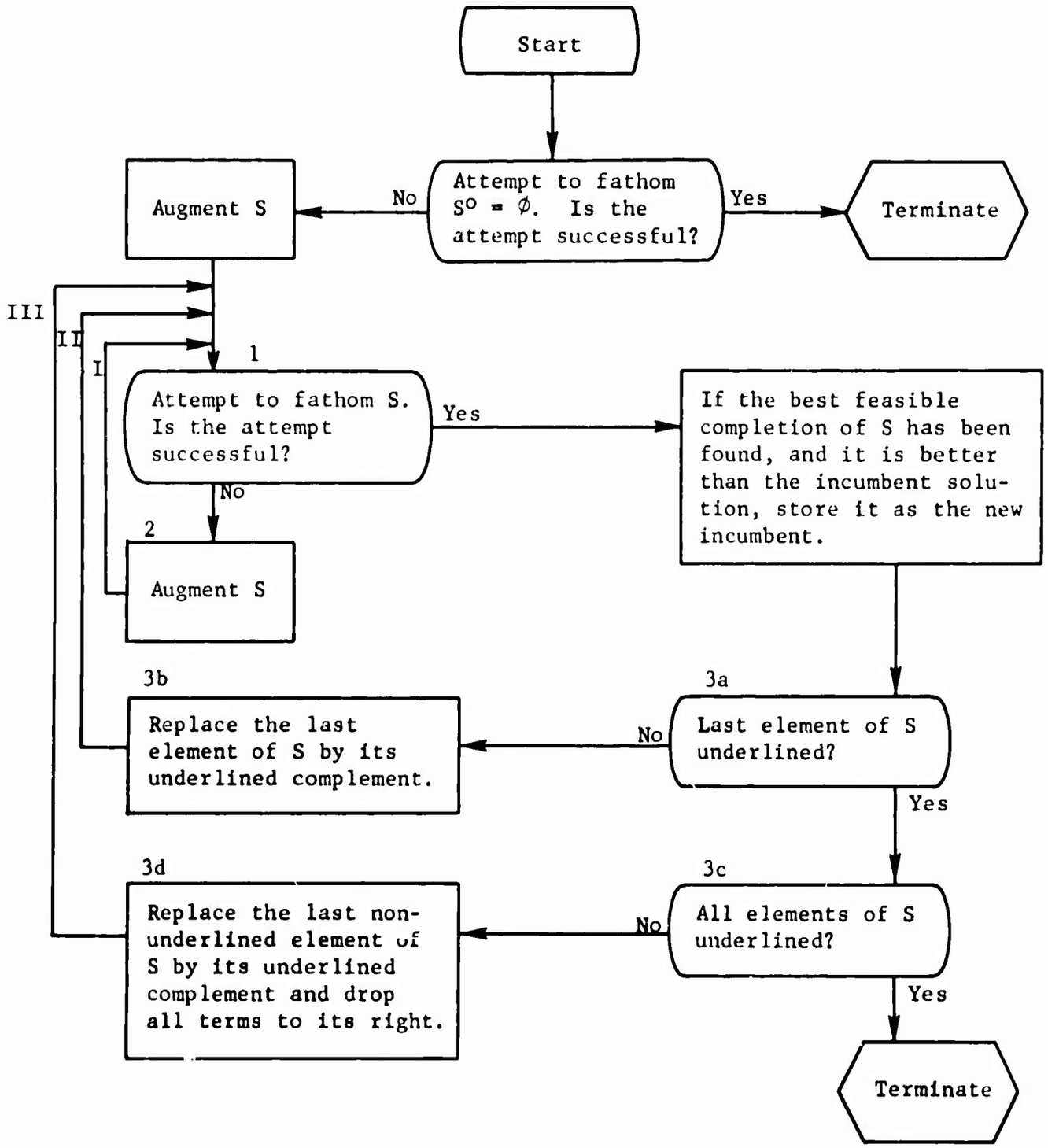

Fig. 1 -- Flow Chart of a General Enumerative Procedure[a]

[a]To facilitate proof of its validity, the scheme is not presented as compactly as it could be.

Theorem. The procedure of Fig. 1 leads to a non-redundant sequence of partial trial solutions which does not terminate before all 2^n solutions have been (implicitly) enumerated.

Before proving this theorem, we note its main consequence: if the mechanisms of Steps 1 and 2 are computationally finite (as they are in Balas' algorithm and certainly would be in any reasonable algorithm that uses the procedure of Fig. 1), then Fig. 1 is computationally finite and an optimal feasible solution of (P) is in store at termination if the collection of feasible solutions is not empty. If there are no feasible solutions, then, of course, no feasible solution is in store at termination.

Proof.* If $S^0 = \phi$ can be fathomed, the theorem is obviously true. Hence, we may assume that ϕ cannot be fathomed.

To show that $< S^\nu >$ is non-redundant, we shall show that if $S^1, \ldots, S^\nu$ is non-redundant, then $S^{\nu+1}$ cannot be redundant; i.e., that $S^{\nu+1}$ must include the complement of at least one element from each of the partial solutions fathomed prior to $S^{\nu+1}$. There are three pathways by which $S^{\nu+1}$ can be determined from S^ν; they are labeled I, II, and III in Fig. 1. If pathway I is taken, the desired conclusion follows from $S^\nu \subset S^{\nu+1}$. If pathway II is taken, the desired conclusion follows from the definition of Step 3b and the implication of the negative branch of test 3a--that S^ν less its last element is not redundant. To establish the desired conclusion for pathway III, we observe from Fig. 1 that the element complemented in Step 3d was contained in every

*Glover[3] has sketched an inductive proof of an essentially equivalent theorem.

partial solution since it was originally introduced (and hence in every fathomed partial solution since that time), and that $S^{\nu+1}$ less its last element is not redundant with respect to the partial solutions (if any) fathomed prior to the time that the deleted element was introduced.

It remains to show that $< S^{\nu} >$ terminates only if all 2^n solutions have been enumerated. From Fig. 1 we see that $< S^{\nu} >$ terminates only if a partial solution consisting of all underlined elements is fathomed. From our earlier remarks concerning the "forgetting of history," we see that the proof would be at hand if we could show that every partial solution S has the property an underlined element implies that all completions of that portion of S up to and including the complement of the underlined element have been enumerated. Now underlined elements have two possible origins: Steps 3b and 3d. Any underlined element created at Step 3b obviously has the asserted property. To see that the same is true of underlined elements created at Step 3d, we begin by considering the first time Step 3a is positive, i.e., the first time Step 3d could occur. Then all underlined elements of S must have been created at Step 3b, and both S and its immediate predecessor must have been fathomed. It follows that S less its right-most consecutive elements has been fathomed--all of the completions of this deleted partial solution have been enumerated. Thus, the new partial solution generated at the first execution of Step 3d has the desired property. Parallel arguments hold for each subsequent execution of 3d. The proof is now complete.

For completeness we note an observation that is clear from the above discussion, and is of interest when (perhaps due to excessive

computation times) the procedure of Fig. 1 is stopped prior to termination. At any iteration, from the current partial solution (including the underlines) it is immediately evident exactly which solutions have been accounted for. For example, if n = 24 and the calculations are stopped when S = $\{5,4,-\underline{2},\underline{3},9\}$, then all $2^{(24-3)}$ completions of $\{5,4,2\}$ have been accounted for, as have all $2^{(24-4)}$ completions of $\{5,4,-2,-3\}$. Thus, it is known precisely which $2^{21} + 2^{20}$ of the 2^{24} solutions have been enumerated, and the current incumbent is an optimal feasible solution of (P) with the additional restrictions, $[x_3 = 1,\ x_4 = 1]$ and either $[x_2 = 1]$ or $[x_2 = 0,\ x_3 = 0]$.

III. A PARTICULARIZATION OF THE PROCEDURE BASED ON BALAS' ALGORITHM

In Sec. II we presented and justified a general enumerative scheme for finding an optimal feasible solution of (P) by implicit enumeration. Details for the mechanisms of Steps 1 and 2 based on Balas' algorithm will now be derived. Remember that many other choices for these mechanisms exist, and that our appeal to Balas' work is mainly illustrative.

Beginning with Step 1, the problem here is to "fathom" the current partial solution, S. Recall that S may be fathomed by doing either of the following:

(i) Finding the best feasible completion of S.

(ii) Determining that no feasible completion of S has a lower value of the objective function than the incumbent.

The general strategy will be to attempt to fathom S by taking each tack in turn by means of very simple computations.

Associated with S is a best (not necessarily unique or feasible) completion x^S of S. Constructing such a best completion is trivial--just take $x_j^S = 0$ or 1 for each free variable according as $c_j \geq 0$ or < 0. For convenience we shall assume without loss of generality* that $c \geq 0$, so that each free variable x_j^S may be taken to be 0. Observe that if x^S is feasible, then x^S is a best feasible completion of S and S is thereby fathomed. Since the computation of x^S is so easy, we shall test its feasibility as the first substep of Step 1 of Fig. 1. As the computations proceed, the value of the incumbent feasible solution gives a (hopefully good) upper bound $\overline{z^*}$ on the optimal value z^* of (P) that can be used to

*If an original c_j were negative, one should make the corresponding substitution $x_j' = 1-x_j$.

good advantage as indicated below. Until the first feasible solution has been found, we take $\overline{z^*} = \infty$.

If the best completion x^s is not feasible, we do nothing further to find the best feasible completion. Instead, we attempt to determine that no feasible completion of S is better than the incumbent. If this is actually the case, then it must be impossible to complete S so as to eliminate all of the infeasibilities of x^s and yet improve upon $\overline{z^*}$. To demonstrate this impossibility, it is clearly sufficient to contemplate non-zero binary values only for the variables in

$$T^S \equiv \{j \text{ free: } cx^s + c_j < \overline{z^*} \text{ and } a_{ij} > 0 \text{ for some } i \text{ such that } y_i^s < 0\},$$

where $y^s = Ax^s + b$, for to give a value of 1 to some free variable not in T^s would either lead to a higher value than $\overline{z^*}$ or would not contribute to diminishing an infeasibility of x^s (we have made use of our assumption that $c \geq 0$). Hence, if T^s is empty then there could be no feasible completion of S that is better than the incumbent, and S is fathomed. It is also easy to see that the same conclusion holds if

$$y_i^s + \sum_{j \in T^s} \max\{0, a_{ij}\} < 0$$

for some i such that $y_i^s < 0$; for then there could be no way to select free variables so as to eliminate infeasibility. So much for Step 1.

For the augmentation mechanism of Step 2, one choice is to augment S by one variable from T^s--the one that leaves the least amount of total infeasibility in the next x^s in the sense of making

$$\sum_{i=1}^{m} \min\{y_i^s + a_{ij}, 0\}$$

an algebraic maximum.

The above details have been incorporated into the procedure of Fig. 1 as Fig. 2. The logic of Fig. 1 has been rearranged for compact presentation, but it should be evident that the logics of Figs. 1 and 2 are in fact equivalent.

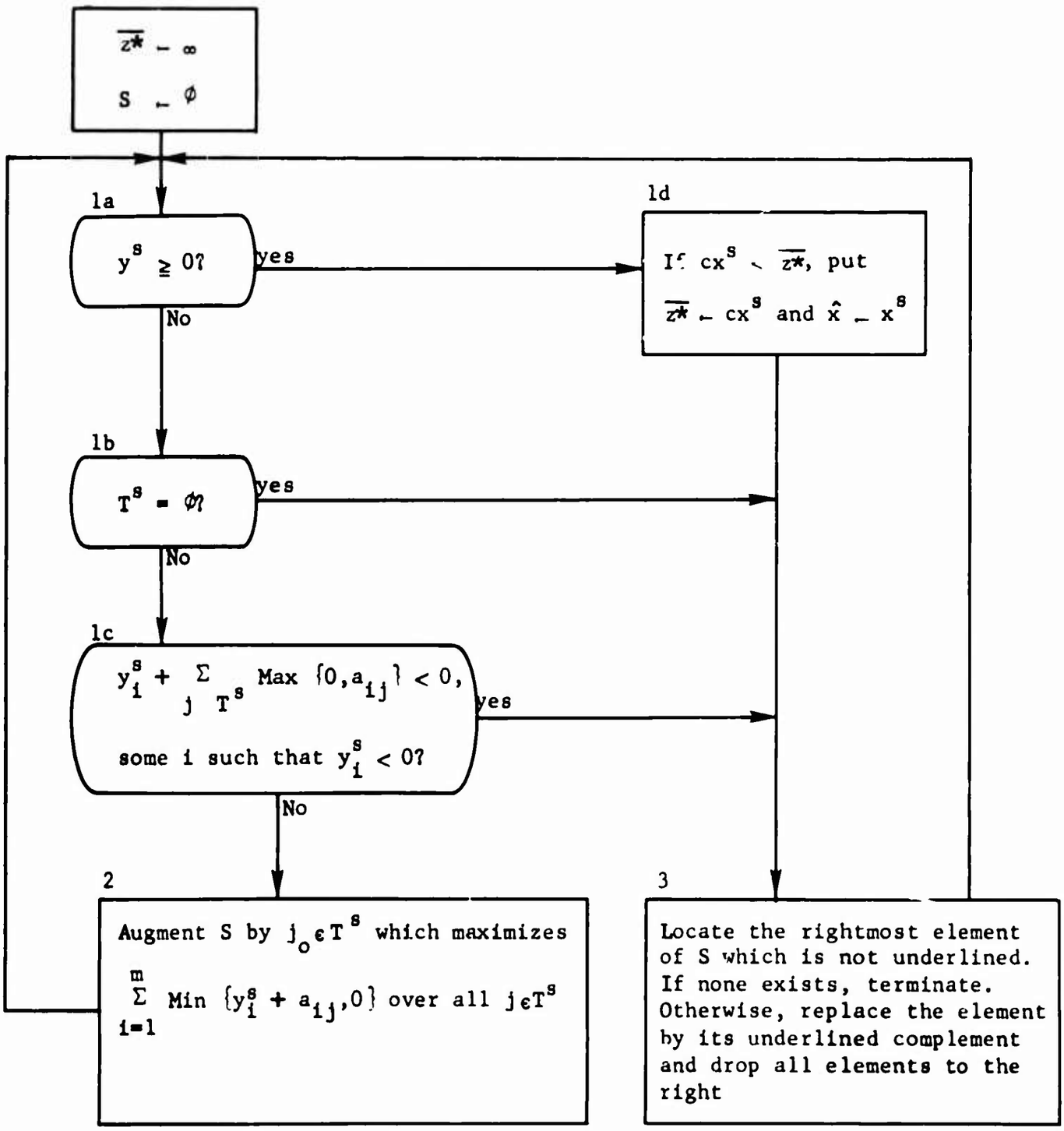

Fig. 2 -- Flow Chart of a Particularization of the Procedure of Fig. 1 Based on Balas' Algorithm

IV. AN EXAMPLE*

Let it be desired to minimize $5x_1 + 7x_2 + 10x_3 + 3x_4 + x_5$ over all binary $x_1,\ldots,x_5$ that satisfy:

$$-2 + x_1 - 3x_2 + 5x_3 + x_4 - 4x_5 \geq 0$$

$$-2x_1 + 6x_2 - 3x_3 - 2x_4 + 2x_5 \geq 0$$

$$-1 - x_2 + 2x_3 - x_4 - x_5 \geq 0 .$$

Summarized below are the calculations that arise from applying the procedure of Fig. 2 to this problem. The subscripts on y and T are suppressed.

Step		
	$\overline{z^*}$	$= \infty$
	S	$= \phi$
1a	y	$= (-2,0,-1) \neq 0$
1b	T	$= \{1,3,4\} \neq \phi$
1c	i	$= 1: \quad -2 + 7 \geq 0$
	i	$= 3: \quad -1 + 2 \geq 0$
2	j	$= 1: \quad -1 - 2 - 1 = -4$
	j	$= 3: \quad -3 \qquad = \boxed{-3}$
	j	$= 4: \quad -1 - 2 - 2 = -5$

Hence,

$S^1 = \{3\}$.

*We have chosen to base our example on a problem taken from Balas[2] so that the interested reader may compare the present algorithm with the original one.

Step

1a $y = (3,-3,1) \not\geq 0$

1b $T = \{2,5\} \neq \phi$

1c $i = 2: \ 5 \geq 0$

2 $j = 2:$ $\boxed{0}$

$j = 5: \ -1 - 1 = -2$

Hence,

$S^2 = \{3,2\}.$

$y = (0,3,0) \geq 0$ (S^2 fathomed)

$cx^s = 17 < \infty$; $\overline{z^*} \leftarrow 17$; $\hat{x} \leftarrow (0,1,1,0,0)$

3 $S^3 = \{3,-\underline{2}\}$

1a $y = (3,-3,1) \not\geq 0$

1b $T = \{5\} \neq \phi$

1c $i = 2: \ -3 + 2 < 0$ (S^3 fathomed)

3 $S^4 = \{-\underline{3}\}$

1a $y = (-2,0,-1) \not\geq 0$

1b $T = \{1,4\} \neq \phi$

1c $i = 1: \ -2 + 1 + 1 = 0$

$i = 3: \ -1 - 1 = -2 < 0$ (S^4 fathomed)

3 Terminate. An optimum solution is (0,1,1,0,0), and the optimal value of the objective function is 17.

V. FURTHER REMARKS

Several further remarks that enhance the usefulness and efficiency of the procedure of Figs. 1 and 2 are now presented.

FINDING ALTERNATIVE OPTIMAL SOLUTIONS

The procedure of Fig. 1 finds exactly one optimal solution of (P) when its constraints are consistent. To find alternative optimal solutions, the procedure can be restarted (see below) with the following changes:

(a) put S^o equal to a permutation of the known optimal solution;

(b) in part (ii) of the definition of fathoming (see Sec. III), replace "a lower" by "at least as low a";

(c) in the step immediately after the positive branch of Step 1, replace "better" by "as good as" and print out each new incumbent.

In terms of Fig. 2, the initial $\overline{z^*}$ should be taken as z^*, and (b) is effectively accomplished by slightly modifying the definition of T^s as follows:

$$T^s = \{j \text{ free: } cx^s + c_j \geq \overline{z^*} \text{ and } a_{ij} > 0 \text{ for some } i \text{ such that } y_i^s < 0\}.$$

It should be noted, however, that not all alternative optima are necessarily found when some $c_j = 0$, for then any corresponding variable that appears with the value 0 in an optimal feasible solution can also be assigned the value 1 without destroying optimality if the resulting solution is still feasible.

USING PRIOR INFORMATION TO MAKE A BETTER START

In many realistic problems, an upper bound on z^* is known a priori. In this case, $\overline{z^*}$ can be initially set at this upper bound

rather than at ∞, with the result that convergence should be speeded up due to greater fathoming ability in the early stages of the computations.

When a feasible solution is known a priori, $\overline{z^*}$ can be put equal to its value, and it is clear from the arbitrary nature of Step 2 that S^o can be initially taken as a permutation of this solution rather than as the empty set. When the feasible solution is a good one, as is likely if it is produced by insight into the problem or the solution to a very similar problem or by a heuristic method, then convergence should be improved.

Still other times, it isclear a priori that certain variables must take on certain values at an optimum solution. Such variables should, of course, be eliminated from the problem statement, as by assigning them the appropriate values and relabeling the remaining variables. Another way of accomplishing essentially the same thing is to take S^o as any permutation of these values underlined. If it is certain that $x_2 = 0$ and $x_7 = 1$ at an optimum, for example, S^o could be taken as $\{\underline{-2},\underline{7}\}$. If it is likely a priori that certain variables take on certain values at an optimum solution, then again by the arbitrary nature of Step 2, S^o can be taken as a permutation of these values. Since the earlier elements of S^o will be complemented during the course of the calculations after the later elements are complemented, by the nature of the basic enumerative scheme it seems reasonable to choose the permutation that ranks the variables in decreasing order of certainty as to their values. For example, it may be deemed "very likely" a priori that $x_7 = 1$ at an optimum, and "fairly likely" that $x_2 = 0$. Then S^o should be taken as $\{7,-2\}$ rather than as $\{-2,7\}$. The same strategy for choosing a permutation can be applied when S^o is to be a known feasible solution.

FINDING ANY FEASIBLE SOLUTION

For some problems, a feasible solution of $b + Ax \geq 0$ is all that is desired. In this case, the objective function is entirely arbitrary, and the calculations can be terminated after the first feasible solution is found. A good choice for c is 0.

NONLINEAR OBJECTIVE FUNCTIONS

The underlying enumerative scheme of Fig. 1 is not in any way dependent on the linear nature of (P). It just happens that in the linear case, reasonably efficient details for Steps 1 and 2 appear to be available. If these details do indeed prove to be efficient, it is natural to want to extend them to integer nonlinear problems. We now indicate how this can be done when the constraints remain linear.

Let it be desired to minimize a nonlinear function, f(x), subject to $b + Ax \geq 0$ and x binary. The crux of the matter is to compute economically, given a partial solution, S, a best completion x^S of S. That is, we must be able to minimize f(x) over the free binary variables while the variables in S are held at their assigned values. If this can be done, it can be shown that the procedure of Fig. 2 remains valid if we make the following modifications: (a) x^S defined as above; (b) T^S defined as $\{j$ free: $f(x^{S/j}) < \overline{z^*}$ and $\tilde{a}_{ij} > 0$ for some $i \ni y_i^S < 0\}$, where $x^{S/j}$ is just x^S with the free variable x_j complemented, and $\tilde{a}_{ij}$ is defined to be a_{ij} if $x_j^S = 0$ and $-a_{ij}$ if $x_j^S = 1$; (c) $\tilde{a}_{ij}$ replaces a_{ij} in Steps 1c and 2; and (d) $f(x^S)$ replaces cx^S in Step 1d.

It should be borne in mind that the computational efficiency of an integer nonlinear programming algorithm of this type is, at this early stage of development, a completely open question.

REFERENCES

1. Balinski, M. L., "Integer Programming: Methods, Uses, Computation," Management Science, Vol. 12, No. 3 (November 1965), pp. 253-313.

2. Balas, E., "An Additive Algorithm for Solving Linear Programs With Zero-One Variables," Operations Research, Vol. 13, No. 4 (July-August 1965), pp. 517-546.

3. Glover, F., "A Multiphase-Dual Algorithm for the Zero-One Integer Programming Problem," Operations Research, Vol. 13, No. 6 (November-December 1965), pp. 879-919.

4. Freeman, R. J., Computational Experience With the Balas Integer Programming Algorithm, The RAND Corporation, P-3241, October 1965.

DOCUMENT CONTROL DATA

1 ORIGINATING ACTIVITY	2a REPORT SECURITY CLASSIFICATION
THE RAND CORPORATION	UNCLASSIFIED
	2b GROUP

3. REPORT TITLE

INTEGER PROGRAMMING BY IMPLICIT ENUMERATION AND BALAS' METHOD

4. AUTHOR(S) (Last name, first name, initial)

Geoffrion, Arthur M.

5. REPORT DATE	6a. TOTAL NO. OF PAGES	6b. NO. OF REFS.
February 1966	28	4

7. CONTRACT or GRANT NO.	8. ORIGINATOR'S REPORT NO.
AF 49(638)-1700	RM-4783-PR

9a AVAILABILITY/LIMITATION NOTICES	9b SPONSORING AGENCY
DDC 1	United States Air Force Project RAND

10. ABSTRACT

A reformulation of the essentials of Balas' algorithm for the zero-one integer linear programming problem. The study is based upon the concept of "elementary tree search" used by Glover as the basis of his multi-phase-dual algorithm. The reformulation requires much less computer storage than the original version; it also clarifies the rationale behind the algorithm, thereby leading naturally to variants and extensions. 28 pp. Illus.

11 KEY WORDS

Integer programming
Algorithms
Linear programming
Operations research
Computer programming